Sub A2

CLAIMS

1. A method of executing a single instruction parallel multiply-add function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel multiply-add instruction;

providing the processor with a first, a second and a third value, wherein each of the values comprises two or more operand components;

multiplying first operand components of the first and the second values to generate a first intermediate value;

multiplying second operand components of the first and the second values to generate a second intermediate value;

adding a first operand component of the third value to the first intermediate value to generate a first result value;

adding a second operand component of the third value to the second intermediate value to generate a second result value;

storing the first result value in a first portion of a result location; and

storing the second result value in a second portion of the result location.

2. The method of claim 1, wherein the first, second and third values are stored in respective source registers of the processor specified by the parallel multiply-add instruction, and the first and the second result values are stored in a destination register of the processor specified by the parallel multiply-add instruction.

A2

3. The method of claim 2, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

4. The method of claim 1, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction every 2 cycles.

5. A method of executing a single instruction conditional pick function on a processor, the method comprising:

providing the processor with an opcode indicating a conditional pick instruction;

providing the processor with a first, a second and a third value;

comparing the first value to a reference value;

determining, based upon the comparing, whether the first value is equal to the reference value;

storing the second value in a result location if the first value is equal to the reference value; and

storing the third value in a result location if the first value is not equal to the reference value.

6. The method of claim 5, wherein the first, second and third values are stored in respective source registers of the processor specified by the conditional pick instruction, and the second and the third values are stored in a destination register of the processor specified by the conditional pick instruction.

7. The method of claim 5, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

8. A method of executing a single instruction parallel averaging function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel averaging instruction;

providing the processor with a first and a second value, wherein each of the values comprises two or more operand components;

adding first operand components of the first and the second values to generate a first intermediate value;

adding second operand components of the first and the second values to generate a second intermediate value;

incrementing the first intermediate value by one to generate a third intermediate value;

incrementing the second intermediate value by one to generate a fourth intermediate value;

shifting the third intermediate value to generate a first result value;

shifting the fourth intermediate value to generate a second result value;

storing the first result value in a first portion of a result location; and

storing the second result value in a second portion of the result location.

9. The method of claim 8, wherein the first and the second values are stored in respective source

registers of the processor specified by the parallel averaging instruction.

10. The method of claim 8, wherein the first and the second result values are stored in a destination register of the processor specified by the parallel averaging instruction.

11. The method of claim 10, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

12. The method of claim 8, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

13. A method of executing a single instruction parallel shift function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel shift instruction;

providing the processor with a first and a second value, wherein each of the values comprises two or more operand components;

shifting the first operand component of the first value by a number of bits equal to a value of the first operand component of the second value to generate a first result value;

shifting the second operand component of the first value by a number of bits equal to a value of the second operand component of the second value to generate a second result value;

storing the first result value in a first portion of a result location; and

storing the second result value in a second portion of the result location.

14. The method of claim 13, wherein the first and the second values are stored in respective source registers of the processor specified by the parallel shift instruction.

15. The method of claim 13, wherein the first and the second result values are stored in a destination register of the processor specified by the parallel shift instruction.

16. The method of claim 15, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

17. The method of claim 13, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

Sub A2

18. A general purpose processor comprising:

a file register;

an instruction fetch unit; and

decoding circuitry;

wherein the processor supports a parallel multiply-add instruction.

19. The general purpose processor of claim 18, wherein the parallel multiply-add instruction operate

A2

on either integer or fixed point operands.

20. The general purpose processor of claim 19, wherein the results of the parallel multiply-add instruction are saturated.

21. The general purpose processor of claim 19, wherein the parallel multiply-add instruction further provides multiple saturation modes.

22. A general purpose processor comprising:
a file register;
an instruction fetch unit; and
decoding circuitry;
wherein the processor supports a conditional pick instruction.

23. A general purpose processor comprising:
a file register;
an instruction fetch unit; and
decoding circuitry;
wherein the processor supports a parallel averaging instruction.

24. A general purpose processor comprising:
a file register;
an instruction fetch unit; and
decoding circuitry;
wherein the processor supports a parallel shift instruction.

A2

25. A general purpose processor comprising:
a file register;
an instruction fetch unit; and

A2

decoding circuitry;

wherein the processor supports a parallel power instruction.

26. A general purpose processor comprising:

a file register;

an instruction fetch unit; and

decoding circuitry;

wherein the processor supports a parallel reciprocal square root instruction.

Add A2